Report: Detecting Cars in Satellite Images

David Tafler

I. Introduction

Prior to the adoption of convolutional neural networks (CNNs), object detection relied heavily on the design of hand-crafted features [ZSGY19]. However, the development of the first CNN-based two-stage detector changed the game by generating bounding box candidates with region-proposal methods, followed by the classification of the box's contents in a second step [GDDM14]. YOLO, on the other hand, conceptualizes object detection as a regression problem, where each grid cell in an image predicts objects centered in it [RDGF16]. The report presents an implementation of YOLO-like object detectors trained to detect cars in satellite images, starting with training a detector on the Cars Without Context (COWC) dataset and then extracting its features to train two further models on images of Stirling (UK) and Würzburg (Germany).

II. Proposed Solution with Justifications

A. Data

The pretraining dataset contained 25384 labeled images from four cities, with at least one car in each image. Two cities with gray-scale images were excluded to make the training process easier, although including them would have helped the models generalize better. This decision reflects the theme of prioritizing early training successes at the expense of model generalizability to achieve initial results within a limited time-frame.

To create the Würzburg and Stirling datasets, I used Google Earth to obtain large images with around 15cm per pixel at ground level, similar to the COWC dataset. The images were then split into smaller ones, with half of each city's images randomly assigned to a training set and the other half to a testing set. I used Label Studio [TMHL20] to annotate the images, resulting in a total of 1259 images.

The data was read using tensorflow.data.Dataset objects, which are efficient due to caching and prefetching capabilities. Images were resized, but no further pre-processing was done as the network can learn any transformation function, and the models rescale pixel values in the first layers. While image augmentation methods could improve performance, the difficulty of automatically augmenting the bounding-box labels made it impractical. Labels were represented as lines of text, where each line describes the location of a bounding-box and its class. Based on this information, the training-labels were created. They represent images as grids and if the center of an object is located within a certain grid-cell, this cell contains information about the class and bounding box coordinates of that object. Relevant conversion functions found in the ”utils.py” file.

B. Model Architectures

The base-model was pretrained on the COWC data-set, utilizing EfficientNetV2S without the classification layer as its base, followed by convolutional blocks with residual connections and a model head. Instead of pooling, the model learns how to downscale through convolution and has fewer parameters since fully connected layers are not used [RF16; SDBR14]. The structure of the model can be seen in Figure 1.

To train the transfer models on the Würzburg and Stirling data-sets, the weights of the pretrained base-model were frozen, and further convolutional-residual blocks were added. Two sets of transfer models were trained: one using later (deeper) features of the base-model and the other using earlier (shallower) features. All models have a similar number of trainable parameters. Although only the late-feature models are included in the submitted notebook, this report provides an overview of the results of all the models.

C. Training

To optimize the model, AMSGrad [RKK19] was employed, a variant of ADAM with “long-term memory”, which did not get stuck in local optima as often as ADAM in my experiments. Keras Tuner with random search found learning rates for pretraining (0.03) and transfer learning (0.09). The base-model was trained for 20 epochs using a training/validation split of 0.1, while each transfer model was trained for 60 epochs. Checkpoints with the lowest validation loss were saved.

III. Results

In Object-detection, metrics such as precision or recall depend on thresholds of confidence and IoU, which makes simple confusion matrices meaningless. Average-Precision (AP) overcomes this issue by computing the area under the precision-recall curve of one class, and mAP is the mean AP of all classes. An open-source implementation was used to calculate mAP [Car18].

Table I presents the mAP results. “Base” denotes the model trained on the COWC training set, while “Würzburg” and “Stirling” refer to models with a pretrained base trained on the respective training data. While the performance of the base model on the COWC set was good (see also Figure 2), it failed

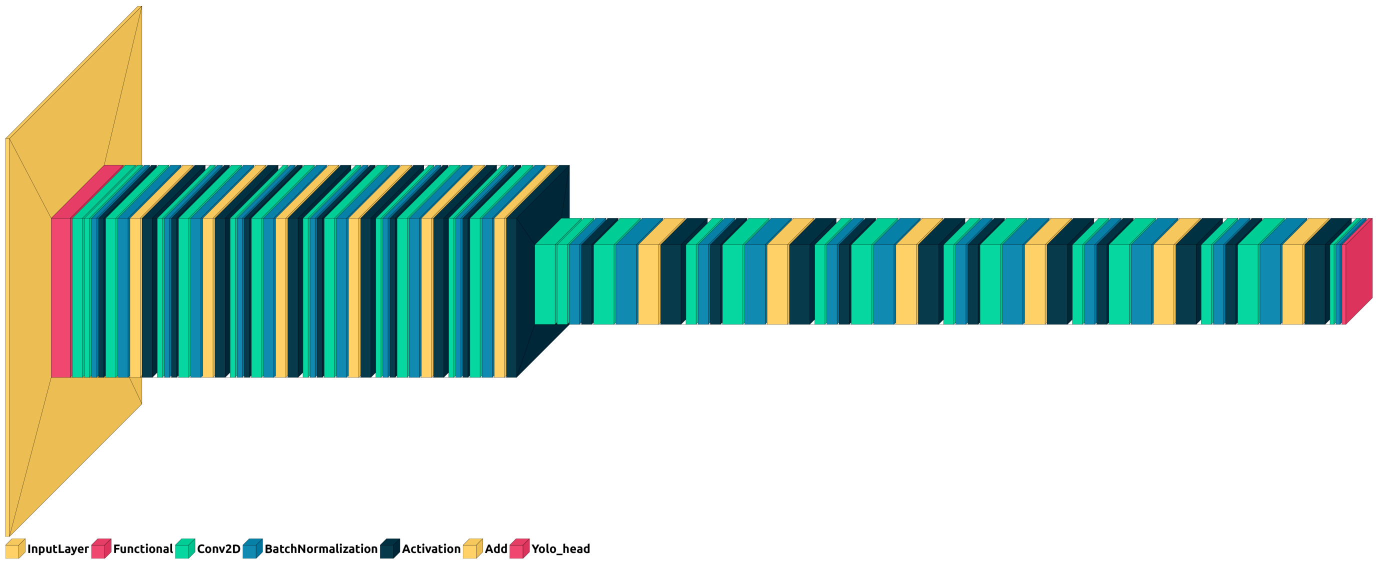

Fig. 1: Visualization of the base model

to generalize to new contexts. It generalized and provided a better base for transfer learning to Würzburg than to Stirling, but transfer-learning was not successful, and with the Stirling late-feature-model, performance even worsened.

TABLE I: AP of different models on the validation set of COWC and the test sets of Würzburg and Stirling

Model	**AP**		
	COWC	***Würzburg***	***Stirling***
Base	97.19%[a]	2.41%	0.33%
Early features			
Würzburg	-	16.32%	13.27%
Stirling	-	0.93%	0.97%
Late features			
Würzburg	-	16.41%	5.98%
Stirling	-	0.07%	0.03%

[a]evaluated on only 200 randomly chosen images from the validation set because of computational resources limitations.

IV. Discussion

Building and training a simplified YOLO-like object detector is no trivial task. The base-model shows that training on lots of data results in good performance on similar data. With fewer data, even overfitting the model is not easy. As an illustration, I trained a base-model from scratch on the Stirling data. Curiously, while the training loss converged to a low value of around 0.0099 (for comparison: 0.002 when trained on COWC), the predictions were off. Figure 3 shows some predictions on training-data. The network seems to have settled in a local optimum. Thus, improvements could include (1) a more exhaustive hyper-parameter search focusing on learning-rates and model architecture, (2) collecting more training data and using data augmentation, (3) improving the loss-function to help optimization avoid local optima.

I expect the following examples of methods to improve performance to a lesser degree. A higher grid-resolution helps to better detect objects that are close together. More bounding-box-predictions per grid-cell with variously shaped anchor-boxes makes training with objects of different shapes and sizes easier. Fine-tuning by unfreezing all weights and training with a very low learning-rate and non-max suppression are expected to bring incremental progress.

V. Conclusion

Although the idea of training and end-to-end neural-network is simple, many challenges arise when a computer-vision task becomes more complicated than for example image classification. If mAP had been the only objective to maximize, using an established implementation of a pretrained object detector like yolov5 [JCS+22] as a base model would have been much easier. But with a focus on learning, I implemented a broad range of concepts and functions myself, ranging from loss-functions and model building-blocks to the visualization of bounding boxes.

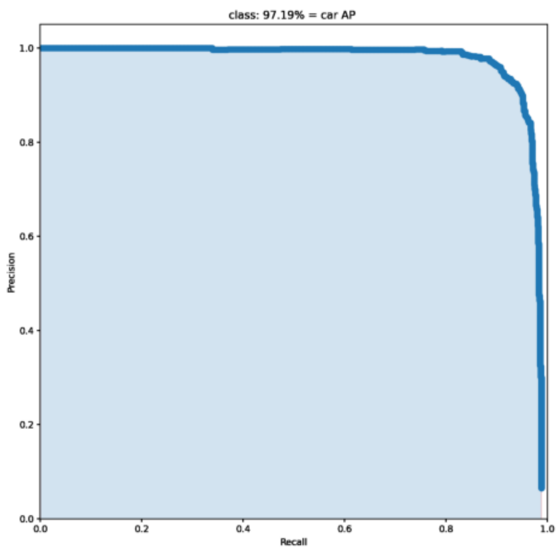

Fig. 2: Precision-recall-curve of the base model on 200 randomly chosen images from the COWC validation set.

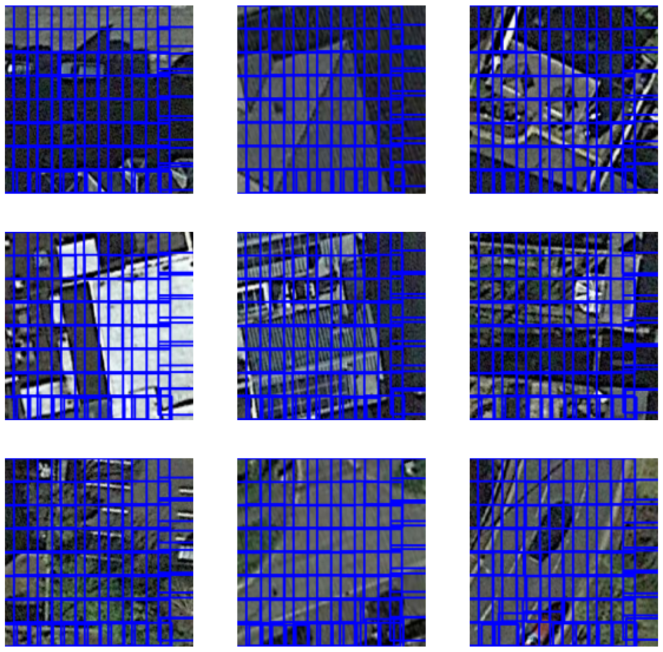

Fig. 3: Predictions with confidence $>$ 0.5 on training data from a base-model trained from scratch on the Stirling training set

References

[Car18] J. Cartucho, *mAP*, 2018. [Online]. Available: https://github.com/Cartucho/mAP.

[GDDM14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *2014 IEEE Conference on Computer Vision and Pat-*

tern Recognition, event-place: Columbus, OH, USA, IEEE, Jun. 2014, pp. 580–587, ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.81. [Online]. Available: http://ieeexplore.ieee.org/document/6909475/ (visited on 10/22/2022).

[JCS+22] G. Jocher *et al.*, *Ultralytics/yolov5: V6.2 - YOLOv5 classification models, apple m1, reproducibility, ClearML and deci.ai integrations*, version v6.2, Aug. 17, 2022. DOI: 10.5281/ZENODO.7002879. [Online]. Available: https://zenodo.org/record/7002879 (visited on 10/22/2022).

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[RF16] J. Redmon and A. Farhadi, *YOLO9000: Better, faster, stronger*, Issue: arXiv:1612.08242, Dec. 25, 2016. arXiv: 1612.08242[cs]. [Online]. Available: http://arxiv.org/abs/1612.08242 (visited on 06/25/2022).

[RKK19] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[SDBR14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[TMHL20] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov, *Label studio: Data labeling software*, 2020. [Online]. Available: https://github.com/heartexlabs/label-studio.

[ZSGY19] Z. Zou, Z. Shi, Y. Guo, and J. Ye, *Object detection in 20 years: A survey*, Issue: arXiv:1905.05055, May 15, 2019. arXiv: 1905.05055[cs]. [Online]. Available: http://arxiv.org/abs/1905.05055 (visited on 10/20/2022).